Nonlinear Systems Solver in Floating-Point Arithmetic using LP Reduction

Christoph Fünfzig, Dominique Michelucci, Sebti Foufou
Laboratoire Le2i, UMR CNRS 5158, Université de Bourgogne
BP 47870, 21078 Dijon Cedex, France
christoph.fuenfzig@u-bourgogne.fr, dmichel@u-bourgogne.fr, sfoufou@u-bourgogne.fr

ABSTRACT

This paper presents a new solver for systems of nonlinear equations. Such systems occur in Geometric Constraint Solving, e.g., when dimensioning parts in CAD-CAM, or when computing the topology of sets defined by nonlinear inequalities. The paper does not consider the problem of decomposing the system and assembling solutions of subsystems. It focuses on the numerical resolution of well-constrained systems. Instead of computing an exponential number of coefficients in the tensorial Bernstein basis, we resort to linear programming for computing range bounds of system equations or domain reductions of system variables. Linear programming is performed on a so called Bernstein polytope: though, it has an exponential number of vertices (each vertex corresponds to a Bernstein polynomial in the tensorial Bernstein basis), its number of hyperplanes is polynomial: $O(n^2)$ for a system in n unknowns and equations, and total degree at most two. An advantage of our solver is that it can be extended to non-algebraic equations. In this paper, we present the Bernstein and LP polytope construction, and how to cope with floating point inaccuracy so that a standard LP code can be used. The solver has been implemented with a primal-dual simplex LP code, and some implementation variants have been analyzed. Furthermore, we show geometric-constraint-solving applications, as well as numerical intersection and distance computation examples.

Categories and Subject Descriptors

G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations; G.1.6 [**Numerical Analysis**]: Optimization; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; J.6 [**Computer-Aided Engineering**]: CAD

General Terms

CAD, Geometric Constraints

Keywords

Geometric Constraint Solving, Subdivision Solver, Intersection Computation, Distance Computation, Interval Arithmetic, Linear Programming

1. INTRODUCTION

All currently available geometric modelers provide possibilities for the modeling of geometric shapes, or the dimensioning of spatial models of CAD parts, using a set of geometric constraints such as incidences, tangencies, angles, and distances [1].

The configuration, which satisfies a set of geometric constraints, can be found by solving a usually big, underlying system of nonlinear algebraic equations. Undecomposable sub-systems may have more than a dozen of unknowns and equations, and are typically solved by numerical methods such as the Newton-Raphson iteration, the continuation method, and their interval variants [11, 4, 20, 10].

The class of subdivision solvers relies on bounds of the system equation values for reducing the domain boxes containing a solution. If a reduction is not efficient then the domain box is subdivided, and the method continues with the sub-boxes [8]. Different approaches have been proposed for bounding the system equation values. The tensorial Bernstein basis (TBB), known from CAGD [6], allows to represent a multivariate polynomial $p(x)$, $x = (x_1, \dots x_n)$ in an n-dimensional rectangular domain $D = \{x \in \mathbb{R}^n | u_i \leq x_i \leq v_i,\ u_i, v_i \in \mathbb{R}\}$. The polynomial values are then enclosed by the interval between the smallest and the largest of the TBB coefficients. The de Casteljau algorithm allows the computation of the TBB coefficients after subdivision into sub-boxes. Sherbrooke and Patrikalakis [17] use the TBB bounds for a subdivision solver, called *interval projected polyhedron algorithm*. Mourrain and Pavone [15] improve the speed of this algorithm by employing efficient univariate solvers and preconditioning. Also the B-spline basis can be used for bounding piecewise polynomial or rational functions, and the paper [5] by Elber and Kim reports on the behavior of the corresponding solver.

A problem of such solvers is that sparse polynomials in the canonical basis $\{x_1^{i_1} \dots x_n^{i_n} : 0 \leq i_1 \leq d_1, \dots 0 \leq i_n \leq d_n\}$ become dense in the TBB [14]. For instance, the monomial 1 is written as

$$1 = (B_0^{(d_1)}(x_1) + \dots + B_{d_1}^{(d_1)}(x_1)) \cdot \dots \cdot (B_0^{(d_n)}(x_n) + \dots + B_{d_n}^{(d_n)}(x_n))$$

in the TBB, where

$$B_i^{(d_k)}(x_k) := \binom{d_k}{i} x_k^i (1-x_k)^{d_k-i}$$

is the i-th polynomial in the Bernstein basis of univariate polynomials of degree d_k. Similarly, a linear polynomial $p(x_1, \dots x_n)$ requires an exponential number of 2^n coefficients in the TBB, while it requires only $n+1$ coefficients in the canonical basis. A quadratic polynomial (where the degree of its monomials is less than or equal to two) requires 3^n coefficients in the TBB and $O(n^2)$ coefficients generated by the canonical basis $\{x_i^2, x_i x_j, x_i, 1\}$, $i \neq j$. This makes Bernstein solvers impractical for systems of more than 6 or 7 equations, and especially for undecomposable systems resulting from geometric constraints. As an example, an icosahedron (20 triangles, 12 vertices, 30 edges) may be specified by the lengths of its edges which gives an undecomposable system of 30 equations. Similarly, the dodecahedron (12 pentagonal faces, 20 vertices, 30 edges) may be specified by the lengths of its 30 edges and the coplanarity of its 12 pentagonal faces. Such systems are quadratic, composed of equations:

$$\begin{aligned} &a_k x_i + b_k y_i + c_k z_i + d_k = 0, \\ &a_k^2 + b_k^2 + c_k^2 = 1, \\ &(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = d_{ij}^2, \end{aligned}$$

where (x_i, y_i, z_i) are the coordinates of vertex i, $a_k x + b_k y + c_k z + d_k = 0$ is the equation of face plane k, and d_{ij} is the length of edge ij. For a well constrained system, three vertices can be fixed (e.g., one vertex at the origin, the second vertex on the x axis, and the third on the plane Oxy).
The resulting systems are big and usually undecomposable. They can not be practically solved using tensorial Bernstein basis solvers because of the exponential number of coefficients in the TBB. The only existing solution without the TBB uses the simplicial (or homogeneous) Bernstein basis instead [16].

This paper proposes a new solver with polynomial time domain reductions, outlined in [12]. It resorts to linear programming for computing domain reductions of system variables or range bounds of system equation. Linear programming is performed using a so called Bernstein polytope: though, it has an exponential number of vertices (each vertex corresponds to a Bernstein polynomial in the TBB), its number of hyperplanes is polynomial. In this paper, we only need to cover quadratic systems, where the degree of each monomial is less than or equal to two. We handle higher degrees by larger systems, which are created in a preprocessing step by repeated substitution of degree-2 factors (Section 6). Then the number of hyperplanes for a quadratic system in n unknowns and equations is $O(n^2)$. In [12], exact rational arithmetic is used. For floating-point arithmetic, we modify the Bernstein polytope's inequalities in dependence of the current domain border inaccuracies in order to not omit solutions. In this way, any fast, standard LP code in floating-point arithmetic can be used.

1.1 Notation

For real intervals occuring in this paper, we use upper case, caligraphic letters. We denote the center of an interval $\mathcal{A}$ by $c(\mathcal{A})$, the lower bound by $u(\mathcal{A})$ or $\underline{\mathcal{A}}$, the upper bound by $v(\mathcal{A})$ or $\overline{\mathcal{A}}$, and the half-width by $r(\mathcal{A}) = v(\mathcal{A}) - c(\mathcal{A})$. Further on, we refer to the domain interval of a real variable x with $\mathcal{D}(x) \subset \mathbb{R}$. For an n-dimensional domain box $\mathcal{D}$, we use $\mathcal{D}_i = [u_i, v_i] \subset \mathbb{R}$ for the domain interval in dimension i, $i = 1, \dots, n$.
Discarding the binomial coefficient in the i-th Bernstein basis polynomial $B_i^{(d_k)}$, $i = 0 \dots d_k$, of degree d_k, gives the i-th unscaled Bernstein basis polynomial, we denote by $\hat{B}_i^{(d_k)}$.

1.2 Paper Outline

Section 2 gives the definition of the Bernstein polytope in a space $\mathbb{R}^N$ using its set of halfspaces. Section 3 shows how the Linear Programming (LP) technique provides bounds for the range of a given multivariate polynomial $p(x), x = (x_1, x_2, \dots x_n)$ and of the domain intervals for $p(x) = 0$. In the subdivision solver, the underlying Bernstein polytope is required for arbitrary domain boxes, and the scaling method in Section 3 details it. It is necessary to cope with floating point inaccuracy in the LP solver, and this is described in Section 4. Finally, we show numerical examples from geometric constraint solving and intersection/distance computation (Section 5). Conclusions and ideas for future improvements follow in Section 6.

2. BERNSTEIN POLYTOPES

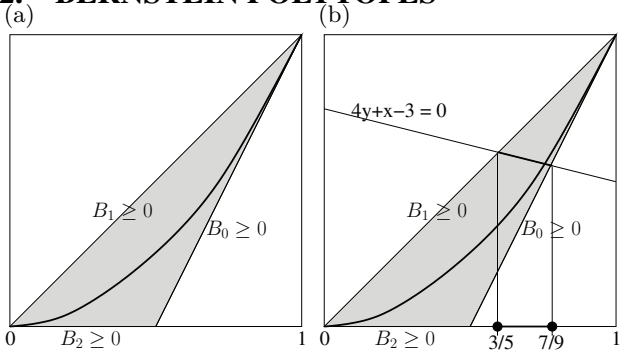


Figure 1: (a) The Bernstein polytope delimiting the curve $(x, y = x^2)$, for $(x, y) \in [0, 1]^2$, is the intersection of the halfspaces $\hat{B}_0^{(2)}(x) = (1-x)^2 = y - 2x + 1 \geq 0$, $\hat{B}_1^{(2)}(x) = 2x(1-x) = 2x - 2y \geq 0$, $\hat{B}_2^{(2)}(x) = x^2 = y \geq 0$. (b) Solving $4x^2 + x - 3 = 0$, $x \in [0, 1]$, turns to computing the intersection of the line $4y + x - 3 = 0$ and the curve (x, x^2). LP computes the intersection of the line and the Bernstein polytope: the initial interval $\mathcal{D}(x) = [0, 1]$ is reduced to $[3/5, 7/9]$.

We delimit the semi-algebraic set

$$S = \{(x_1, x_2, \dots x_n, x_1^2, x_1 x_2, \dots x_1 x_n, \dots x_n^2) : x \in [0, 1]^n\}$$

by a polytope in $[0, 1]^N$, $N = n(n+3)/2$. The polytope is defined by its halfspace inequalities. The set S and this hull polytope belong to an N-dimensional space, which has one dimension for each monomial (except for the monomial 1). We call it the *Bernstein polytope* because its halfspaces are given by the inequalities:

$$\hat{B}_i^{(2)}(x_k) \geq 0 \text{ for } i = 0, 1, 2$$

and

$$\hat{B}_i^{(1)}(x_k) \hat{B}_j^{(1)}(x_l) \geq 0 \text{ with } i = 0, 1,\ j = 0, 1,\ k \neq l.$$

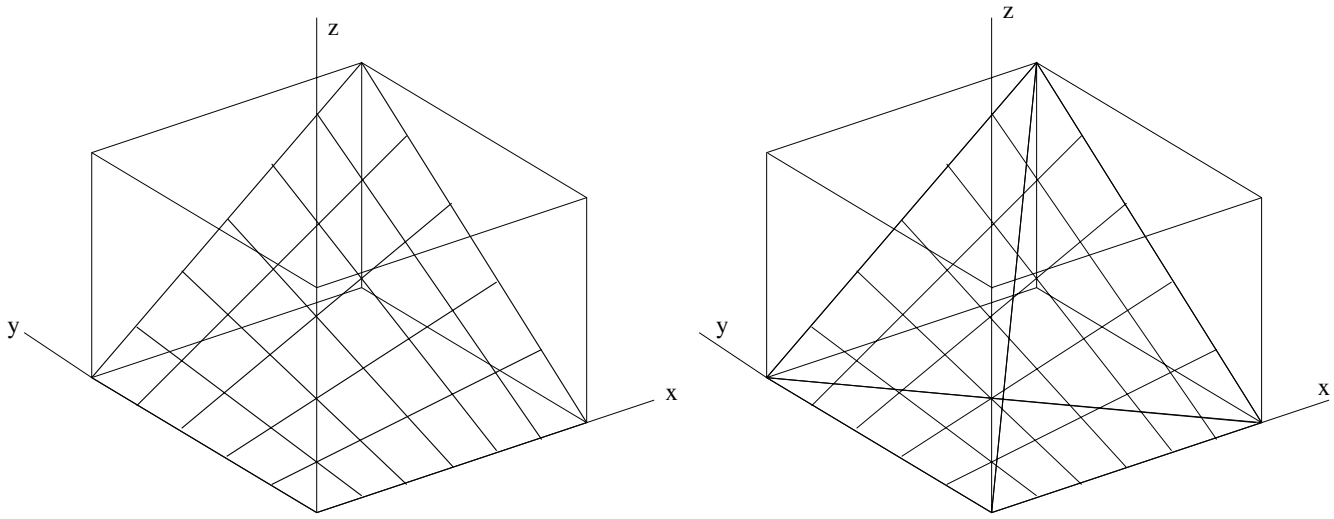


Figure 2: The Bernstein polytope delimiting a patch of the hyperbolic paraboloid: $(x, y, z = xy)$. Inequalities of the four bounding hyperplanes are $\hat{B}_i^{(1)}(x)\hat{B}_j^{(1)}(y) \geq 0$, $i = 0, 1$, $j = 0, 1$ with $\hat{B}_0^{(1)}(t) = 1-t$, $\hat{B}_1^{(1)}(t) = t$, containing three box intersection points each.

We associate with each monomial x_i, x_i^2, $x_i x_j$, $i \neq j$ a new variable X_i, X_{ii}, X_{ij} of an LP problem. For simplicity in this section, we consider only the case x_i in interval $[0, 1]$. In the solver, we need to handle the case x_i in an arbitrary interval $\mathcal{D}_i$. For the arbitrary case, we scale the polytope as described in Section 3.

Let us first detail the polytope's halfspaces. For a variable x_i, the curve segment (x_i, x_i^2) with $0 \leq x_i \leq 1$ is delimited by a triangle, see Figure 1. Let X_{ii} be the variable which represents the monomial x_i^2. The inequalities that define this triangle are

$$\begin{aligned}
&\hat{B}_0^{(2)}(x_i) = (1 - x_i)^2 = x_i^2 - 2x_i + 1 \geq 0 \\
&\quad \rightarrow X_{ii} - 2X_i + 1 \geq 0 \\
&\hat{B}_1^{(2)}(x_i) = 2x_i(1 - x_i) = -2x_i^2 + 2x_i \geq 0 \\
&\quad \rightarrow -2Xii + 2X_i \geq 0 \\
&\hat{B}_2^{(2)}(x_i) = x_i^2 \geq 0 \\
&\quad \rightarrow X_{ii} \geq 0
\end{aligned} \tag{1}$$

The number of these inequalities is $3n$.

For two distinct variables x_i and x_j, the monomial $x_i x_j$ is represented by a variable X_{ij}. In the subspace x_i, x_j, x_{ij}, the surface patch $\{(x_i, x_j, z = x_i x_j) : x_i \in [0, 1], x_j \in [0, 1]\}$ is a hyperbolic paraboloid, whose convex hull is a tetrahedron, (Figure 2). The inequalities that define this tetrahedron are

$$\begin{aligned}
&\hat{B}_0^{(1)}(x_i)\hat{B}_0^{(1)}(x_j) = 1 - x_i - x_j + x_i x_j \geq 0 \\
&\quad \rightarrow 1 - X_i - X_j + X_{ij} \geq 0 \\
&\hat{B}_0^{(1)}(x_i)\hat{B}_1^{(1)}(x_j) = x_j - x_i x_j \geq 0 \\
&\quad \rightarrow X_j - X_{ij} \geq 0 \\
&\hat{B}_1^{(1)}(x_i)\hat{B}_0^{(1)}(x_j) = x_i - x_i x_j \geq 0 \\
&\quad \rightarrow X_i - X_{ij} \geq 0 \\
&\hat{B}_1^{(1)}(x_i)\hat{B}_1^{(1)}(x_j) = x_i x_j \geq 0 \\
&\quad \rightarrow X_{ij} \geq 0
\end{aligned} \tag{2}$$

The number of these inequalities is $4n(n-1)/2$.

Altogether, the Bernstein polytope has a total of $3n+2n(n-1) = n(2n+1)$ hyperplanes in a $N = n(n+3)/2$-dimensional space. In terms of the number n of unknowns in the system of equations, this is a polynomial number of $O(n^2)$ hyperplanes. The number of vertices is exponential in the multivariate case ($n > 1$). The projection of the Bernstein polytope to the space $\{(X_1, X_2, \ldots, X_n)\}$ is a polytope with 2^n vertices (the hypercube $[0, 1]^n$). Consequently, the Bernstein polytope has at least the same number of vertices as the hypercube because the projection of a polytope can not have more vertices than the initial polytope itself.

In practice, other halfspaces can be used to shrink the polytope further. For every pair of distinct variables x_i, x_j, the inequality $(x_i - x_j)^2 \geq 0 \rightarrow X_{ii} + X_{jj} - 2X_{ij} \geq 0$ can be added. The inequality $(x_i - 1/2)^2 \geq 0 \rightarrow X_{ii} - X_i + 1/4 \geq 0$ also further truncates the Bernstein polytope, as shown in Figure 3.

The full list of halfspace inequalities is given below

$0 \leq X_{ii} - 2X_i + 1$	$1 \leq i \leq n$	triangle
$0 \leq -2X_{ii} + 2X_i$	$1 \leq i \leq n$	triangle
$0 \leq X_{ii}$	$1 \leq i \leq n$	triangle
$0 \leq 1 - X_i - X_j + X_{ij}$	$1 \leq i < j \leq n$	tetrahedron
$0 \leq X_i - X_{ij}$	$1 \leq i < j \leq n$	tetrahedron
$0 \leq X_j - X_{ij}$	$1 \leq i < j \leq n$	tetrahedron
$0 \leq X_{ij}$	$1 \leq i < j \leq n$	tetrahedron
$0 \leq X_{ii} + X_{jj} - 2X_{ij}$	$1 \leq i < j \leq n$	auxilliary
$0 \leq X_{ii} - X_i + 1/4$	$1 \leq i \leq n$	auxilliary

For $n = 2$ unknowns, the volume of the polytope is approximately 0.007, while the volume of the hypercube in $N = 5$ dimensions is 1. This ratio decreases exponentially when n increases. The small volume of the polytope explains the efficiency of the solver compared to solvers based on naive interval arithmetic, which delimit S by a polytope much larger than the Bernstein polytope, for instance the hypercube.

The Bernstein polytope can be generalized to higher degrees

$d > 2$. For monomials of total degree d, the polytope has $O(n^d)$ hyperplanes and an exponential number of vertices in the multivariate case, at least 2^n. Figure 4 shows the degree-3 Bernstein polytope for the univariate case: a tetrahedron.

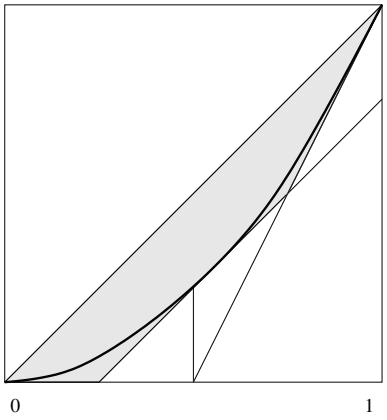


Figure 3: The Bernstein polytope can be shrunk further, e.g., with the inequality $x - y \leq 1/4$.

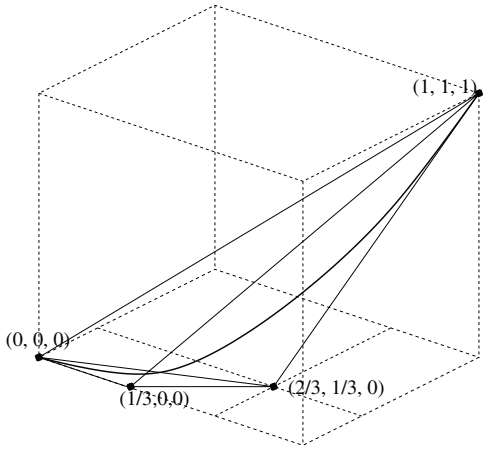


Figure 4: The Bernstein polytope, a tetrahedron, delimiting the curve $(x, y = x^2, z = x^3)$, $x \in [0, 1]$. The polytope vertices are: $v_0 = (0, 0, 0)$, $v_1 = (1/3, 0, 0)$, $v_2 = (2/3, 1/3, 0)$ and $v_3 = (1, 1, 1)$. v_0 is on the plane $\hat{B}_1^{(3)} = \hat{B}_2^{(3)} = \hat{B}_3^{(3)} = 0$, v_1 on the plane $\hat{B}_0^{(3)} = \hat{B}_2^{(3)} = \hat{B}_3^{(3)} = 0$, etc. The tetrahedron is the intersection of four halfspaces $\hat{B}_0^{(3)}(x) = (1-x)^3 = 1-3x+3x^2-x^3 \geq 0$, $\hat{B}_1^{(3)}(x) = 3x(1-x)^2 = 3x - 6x^2 + 3x^3 \geq 0$, $\hat{B}_2^{(3)}(x) = 3x^2(1-x) = 3x^2 - 3x^3 \geq 0$, $\hat{B}_3^{(3)}(x) = x^3 \geq 0$.

3. SOLVER ALGORITHM

For a given quadratic system, we have an enclosure of the semi-algebraic set S by its Bernstein polytope, as detailed in Section 2. The same substitution of monomials x_i by variable X_i, x_i^2 by variable X_{ii}, and $x_i x_j$ by variable X_{ij}, $i \neq j$ gives a *system polytope* of the given quadratic system. For the univariate example in Figure 1, $4x_1^2 + x_1 - 3 = 0$ results in the system polytope $4X_{11} + X_1 - 3 = 0$. Both the system polytope and the Bernstein polytope together, we call the *LP polytope*. Using this LP polytope, we can optimize arbitrary linear objective functions by *linear programming (LP)* [2]. In practice, the simplex algorithm is a good solver for LP problems although it is not polynomial time in the worst case (Klee-Minty examples) [2].

On the LP polytope, we can compute a value bound for one system row by using the system row (after rewriting into LP variables) as objective function. In the example above, minimizing and maximizing $4X_{11} + X_1 - 3$ gives the value bound $[-3, 2]$. Similarly, we can compute domain bounds by using the LP variables X_i, $i = 1, \ldots, n$ as objective functions. In the example above, minimizing and maximizing X_1 results in the domain bound $[3/5, 7/9]$.

The solver manages a stack of domain boxes, which is processed until it is empty. Initially, we start with a huge bounding box D of the solution domain.

Scaling First, we have to scale the Bernstein polytope to the current domain box D. The system polytope's equations are left unchanged by this. For a box $\mathcal{D}(x_i) = [u_i, v_i]$, the inequalities (1)(2) of the polytope halfspaces are modified as follows

$$\begin{aligned}
&\hat{B}_0^{(2)}(x_i) = (v_i - x_i)^2 = x_i^2 - 2v_i x_i + v_i^2 \geq 0 \\
&\quad \rightarrow X_{ii} - 2v_i X_i + v_i^2 \geq 0 \\
&\hat{B}_1^{(2)}(x_i) = 2(x_i - u_i)(v_i - x_i) \geq 0 \\
&\quad \rightarrow 2(-X_{ii} + (u_i + v_i)X_i - u_i v_i) \geq 0 \\
&\hat{B}_2^{(2)}(x_i) = (x_i - u_i)^2 \geq 0 \\
&\quad \rightarrow X_{ii} - 2u_i X_i + u_i^2 \geq 0
\end{aligned} \tag{3}$$

$$\begin{aligned}
&\hat{B}_0^{(1)}(x_i)\hat{B}_0^{(1)}(x_j) = (v_i - x_i)(v_j - x_j) \geq 0 \\
&\quad \rightarrow X_{ij} - v_i X_j - v_j X_i + v_i v_j \geq 0 \\
&\hat{B}_0^{(1)}(x_i)\hat{B}_1^{(1)}(x_j) = (v_i - x_i)(x_j - u_j) \geq 0 \\
&\quad \rightarrow -X_{ij} + u_j X_i + v_i X_j - v_i u_j \geq 0 \\
&\hat{B}_1^{(1)}(x_i)\hat{B}_0^{(1)}(x_j) = (v_j - x_j)(x_i - u_i) \geq 0 \\
&\quad \rightarrow -X_{ij} + u_i X_j + v_j X_i - v_j u_i \geq 0 \\
&\hat{B}_1^{(1)}(x_i)\hat{B}_1^{(1)}(x_j) = (x_i - u_i)(x_j - u_j) \geq 0 \\
&\quad \rightarrow X_{ij} - u_j X_i - u_i X_j + u_i u_j \geq 0
\end{aligned} \tag{4}$$

Similar changes apply to the auxilliary inequalities.

Reduction By solving the LP for the minimum value and the maximum value of the variable X_i, we can reduce the domain interval $\mathcal{D}(x_i)$ in the current LP polytope. See Figure 1(b) for an univariate example. If the LP polytope is infeasible then the studied box does not contain any solution. It is discarded.

Termination If the domain box is smaller than a minimum interval width δ in each dimension, then this domain box potentially contains a solution. It is output. Any termination test, deciding a box has at most one solution, can be included here.

Bisection If the domain box does not reduce by a constant factor (e.g., 0.5) in each reduction step then there might be several solutions. In this case, the domain box is bisected into two sub-boxes, e.g., by bisecting the longest dimension. Several different reduction/bisection orders are possible, see Section 5!

4. COPING WITH FLOATING-POINT INACCURACY

The Bernstein polytope encloses very tightly the underlying semi-algebraic set S. Thus with a naive implementation in floating-point arithmetic, roots can be missed because of rounding errors. For instance, when solving $x_i^2 - x_i = 0$ with $x \in [0,1]$, the line $X_{ii} - X_i = 0$ is considered [12]. If this line becomes $X_{ii} - X_i = \epsilon$ with $\epsilon > 0$ due to inaccuracy, the two roots are missed.

The conceptually simplest solution resorts to exact rational arithmetic [12]. But due to an ever increasing representation size of rational numbers, the solver becomes slower by several orders of magnitude.

The second solution is to use floating point arithmetic. When solving the LP given in Section 2, the LP solver works through non-optimal polytope vertices until it reaches an optimal vertex. Each polytope vertex is defined by a basis matrix selected from the system matrix, which is used to compute all variables (vertex coordinates) and all dual variables (slack variables). The revised simplex method solves these linear systems by forward-backward substitution using a LU decomposition of the basis matrix [2]. The LU decomposition is updated in each basis exchange via a rank-1 update like Forrest-Tomlin [7] and is recomputed after a fixed number of such updates. Solving a linear system $Ax = b$ by forward-backward substitution using a LU decomposition $A + \delta A = LU$, incures rounding errors δx in the solution $x + \delta x$. We bound the relative error $\epsilon(x) := |\delta x|/|x|$ of the solution vector x by backwards error estimation using the matrix condition

$$\begin{array}{l} \epsilon(x) \quad \leq \quad \frac{\kappa(A)}{1-\kappa(A)\epsilon(A)}(\epsilon(A)+\epsilon(b)) \text{ if } \kappa(A)\epsilon(A) < 1 \\ \text{where} \\ \epsilon(A) := |\delta A|/|A| \text{ with a compatible matrix norm,} \\ \kappa(A) := |A||A^{-1}| \text{ is the condition number of } A, \\ \epsilon(b) := |\delta b|/|b| \text{ is the rel. error of the right-hand side.} \end{array} \tag{5}$$

For the absolute error $|\delta x|$ in the maximum norm, we then use the bound with $\epsilon(b) = \epsilon$, the relative machine accuracy, and the matrix condition $\kappa(A)$ computed directly from A and its LU decomposition

$$|\delta x| \quad \leq \quad \max\left(\frac{\kappa(A)}{1-\kappa(A)\epsilon(A)}(\epsilon(A)+\epsilon)|x|,\ \epsilon\right) \tag{6}$$

Finally, a bound for $\epsilon(A) = |\delta A|/|A|$ has been derived by Wilkinson [21, 3]

$$\begin{array}{l} |\delta A_{ij}| \quad \leq \quad 5.01\epsilon\, n \max_k |A_{ij}^{(k)}| \\ \text{where} \\ A_{ij}^{(k)} \text{ is the pivoting element in the } k\text{-th iteration,} \\ \epsilon \text{ is the relative machine accuracy,} \\ n \text{ is the matrix dimension.} \end{array} \tag{7}$$

Using the bounds (6)-(7), we can compute an interval with

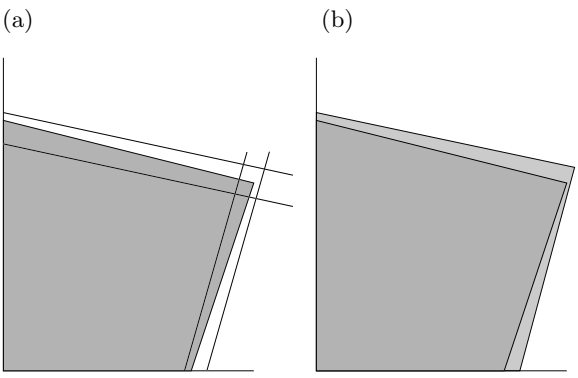


Figure 5: (a) Each bounding line of a 2D convex are sandwiched between two parallel lines (which are not exactly parallel to the exact sandwiched line). (b) Superposition of the exact polygon and its outer approximation. The thickness of the sandwich is exaggerated.

inaccurate borders $\mathcal{D}(x_i) = [[\underline{u_i}, \overline{u_i}]; [\underline{v_i}, \overline{v_i}]]$ after each LP reduction (Section 3). We have to account for these inaccuracies when setting up the LP system in the next iteration. As the Bernstein polytope is of a fixed and simple form, we prefer to scale it (instead of scaling the system polytope as done in [12]).

Geometrically, we extend the feasible set of the Bernstein polytope marginally by pushing hyperplanes outwards as shown in Figure 5. Algebraically, the inaccuracy of coefficients in each row is collected and stored in the column of the constants. Thus, only one column of the polytope contains intervals.

After plugging in the inaccurate borders of $\mathcal{D}(x_i)$, the three inequalities for X_{ii} have the form ($s \in \{+1, -1\}$, $\mathcal{A}, \mathcal{C}$ intervals)

$$\begin{array}{lcl} sX_{ii} + \mathcal{A}X_i + \mathcal{C} & \geq & 0 \\ sX_{ii} + c(\mathcal{A})X_i + ([-r(\mathcal{A}), r(\mathcal{A})]\mathcal{D}(x_i) + \mathcal{C}) & \geq & 0 \\ \text{thick line in } (X_i, X_{ii}) & & \end{array} \tag{8}$$

Similarly, the four inequalities for X_ij have the form ($s \in \{+1, -1\}$, $\mathcal{A}, \mathcal{B}, \mathcal{C}$ intervals)

$$\begin{array}{lcl} sX_{ij} + \mathcal{A}X_j + \mathcal{B}X_i + \mathcal{C} & \geq & 0 \\ sX_{ij} + c(\mathcal{A})X_j + c(\mathcal{B})X_i & & \\ +([-r(\mathcal{A}), r(\mathcal{A})]\mathcal{D}(x_j) + [-r(\mathcal{B}), r(\mathcal{B})]\mathcal{D}(x_i) + \mathcal{C}) & \geq & 0 \\ \text{thick plane in } (X_i, X_j, X_{ij}) & & \end{array} \tag{9}$$

Since all halfspaces are bounded from below, only the lower bound of the interval is relevant.

$$\begin{array}{lcl} sX_{ii} + c(\mathcal{A})X_i & \geq & u(-[-r(\mathcal{A}), r(\mathcal{A})]\mathcal{D}(x_i) - \mathcal{C}) \\ sX_{ij} + c(\mathcal{A})X_j + c(\mathcal{B})X_i & \geq & u(-[-r(\mathcal{A}), r(\mathcal{A})]\mathcal{D}(x_j) \\ & & -[-r(\mathcal{B}), r(\mathcal{B})]\mathcal{D}(x_i) \\ & & -\mathcal{C}) \end{array} \tag{10}$$

We evaluate (10) with interval arithmetic. In this way, we are able to set up the inequalities for an LP solver using standard floating-point arithmetic.

Eventually, it can happen that the Bernstein polytope for X_{ii} (Figure 1) is fully contained in the thick line $-X_{ii} + 0.5(u_i + v_i)X_i + [-\epsilon_i, \epsilon_i][u_i, v_i] - [\underline{u_i}, \overline{u_i}][\underline{v_i}, \overline{v_i}]) \geq 0$, $\epsilon_i := r(0.5([\underline{u_i}, \overline{u_i}] + [\underline{v_i}, \overline{v_i}]))$ corresponding to $B_1^{(2)}(x_i) \geq 0$. In this case, it is wasted resources to represent it by three inequalities. Instead the solver can switch to a thick line, represented by only two inequalities

$$\begin{aligned} -X_{ii} + 0.5(u_i + v_i)X_i &\geq u(-[-\epsilon_i, \epsilon_i][u_i, v_i] \\ &\quad +[\underline{u_i}, \overline{u_i}][\underline{v_i}, \overline{v_i}])) \\ -X_{ii} + 0.5(u_i + v_i)X_i &\leq v(-[-\epsilon_i, \epsilon_i][u_i, v_i] \\ &\quad +[\underline{u_i}, \overline{u_i}][\underline{v_i}, \overline{v_i}])). \end{aligned}$$

In the similar situation for X_{ij} (Figure 2, four inequalities), the solver can switch to a thick plane, represented by only two inequalities

$$\begin{aligned} -X_{ij} - v_i X_j - v_j X_i &\geq u(+[-\epsilon(v_i), \epsilon(v_i)][\underline{u_j}, \overline{v_j}] \\ &\quad +[-\epsilon(v_j), \epsilon(v_j)][\underline{u_i}, \overline{v_i}] \\ &\quad -[\underline{v_j}, \overline{v_j}][\underline{v_i}, \overline{v_i}]) \\ -X_{ij} - v_i X_j - v_j X_i &\leq v(+[-\epsilon(v_i), \epsilon(v_i)][\underline{u_j}, \overline{v_j}] \\ &\quad +[-\epsilon(v_j), \epsilon(v_j)][\underline{u_i}, \overline{v_i}] \\ &\quad -[\underline{v_j}, \overline{v_j}][\underline{v_i}, \overline{v_i}]). \end{aligned}$$

We show results of this switching to thick hyperplanes in Section 5 on numerical examples.

5. NUMERICAL EXAMPLES

In this section, we show and comment on the behavior of the solver on systems of total degree two. All of these systems have been solved using the SoPlex 1.4.1 revised primal-dual simplex code [22]. The implementation has been somewhat optimized by including only the Bernstein inequalities for monomials x_i^2 and $x_i \cdot x_j$ actually used in the given system (*sparse LP polytope*). Furthermore, we use a procedurally generated start basis, which corresponds to the polytope vertex ($X_i = u_i, X_{ii} = u_i^2, X_{ij} = u_i u_j$), $j \neq i$ for minimizing variable X_i, and ($X_i = v_i, X_{ii} = v_i^2, X_{ij} = v_i v_j$), $j \neq i$ for maximizing variable X_i. Note that these vertices are not necessarily feasible for the system polytope but the SoPlex code can switch between the primal (feasible, not optimal) and the dual (optimal, not feasible) simplex algorithm. When solution times are given, they were acquired on a Windows XP 32Bit system (2GB RAM) with an Intel T7200 Core2 Duo processor (2GHz).

5.1 Cone Sections (2D)

On examples in 2-space, i.e., systems with 2 unknowns and 2 equations, we compare the behavior of the solver using LP reductions with a solver using preconditioning and reductions with standard interval arithmetic.

Figure 7 shows the solver using interval arithmetic, and Figure 6 shows the behavior of the solver using LP reductions. All domain boxes are drawn as rectangles inside the starting domain box. The solution points are represented by small black disks, visible inside the smallest boxes. It is immediately visible that the reductions by interval arithmetic are not as efficient, even after preconditioning, and consequently a larger number of them is required. Interval evaluation of the preconditioned system is fast but preconditioning requires a linear system solution (e.g., by LU decomposition), which is a $O(n^3)$ algorithm.

With LP reductions, the smallest visible domain box is reduced in one step to a box smaller than the black disc and therefore can not be seen in the figure. This illustrates the super-quadratic convergence of the solver near regular solutions. In case of a singular solution (the point of intersection between two conics tangent to each other), convergence is not quadratic anymore but the reductions are efficient enough so that the box is not bisected. Infeasible domain boxes are detected very quickly.

5.2 Tangent Line to Four Spheres (3D)

A well constrained problem is finding the lines tangent to four given spheres in 3-space [9]. As unknowns, the system uses the components of the contact points $b_i = (x_i, y_i, z_i)$, $i = 1, 2, 3, 4$ on the four spheres (4 equations). The contact points are collinear (4 equations), and the contact point differences $b_i - b_0$, $i = 1, 2, 3$ are orthogonal to the radius vectors (4 equations). The line is not made explicit in the system but it turns out to be the z-axis in our example, as shown in Figure 8.

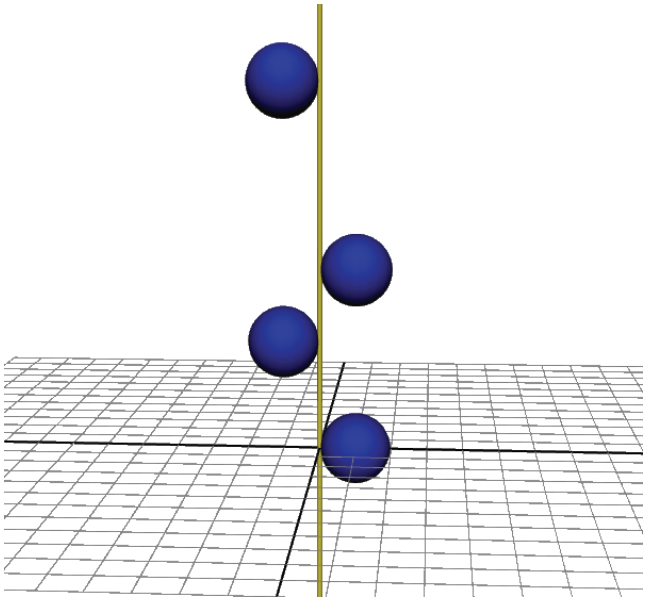


Figure 8: Line tangent to four given spheres (centers $(-1, 0, 0)$, $(1, 0, 3)$, $(-1, 0, 5)$, $(+1, 0, 10)$ and radii $r_i = 1$) in 3-space.

A projection of the analyzed boxes into the x_0, y_0-plane is shown in Figure 9 for different reduction/bisection orders (minimum interval width $\delta = 10^{-2}$). We call a reduction effective if it reduces the interval width by a constant factor 0.5 or it proves the system infeasible. Reduce all dimensions once before bisecting the largest dimension requires 3156 reductions (695 effective) and 287 bisections. Reduce all dimensions as long as effective then bisect the largest dimension does 3119 reductions (539 effective) and 287 bisections. Finally, the strategy: reduce only the largest dimension as long as effective then bisect it, requires 5628 reductions (3141 effective) and a larger number of 2707 bisections. The corresponding solution times are 1.6837 s, 1.6579 s, and 2.5921 s.

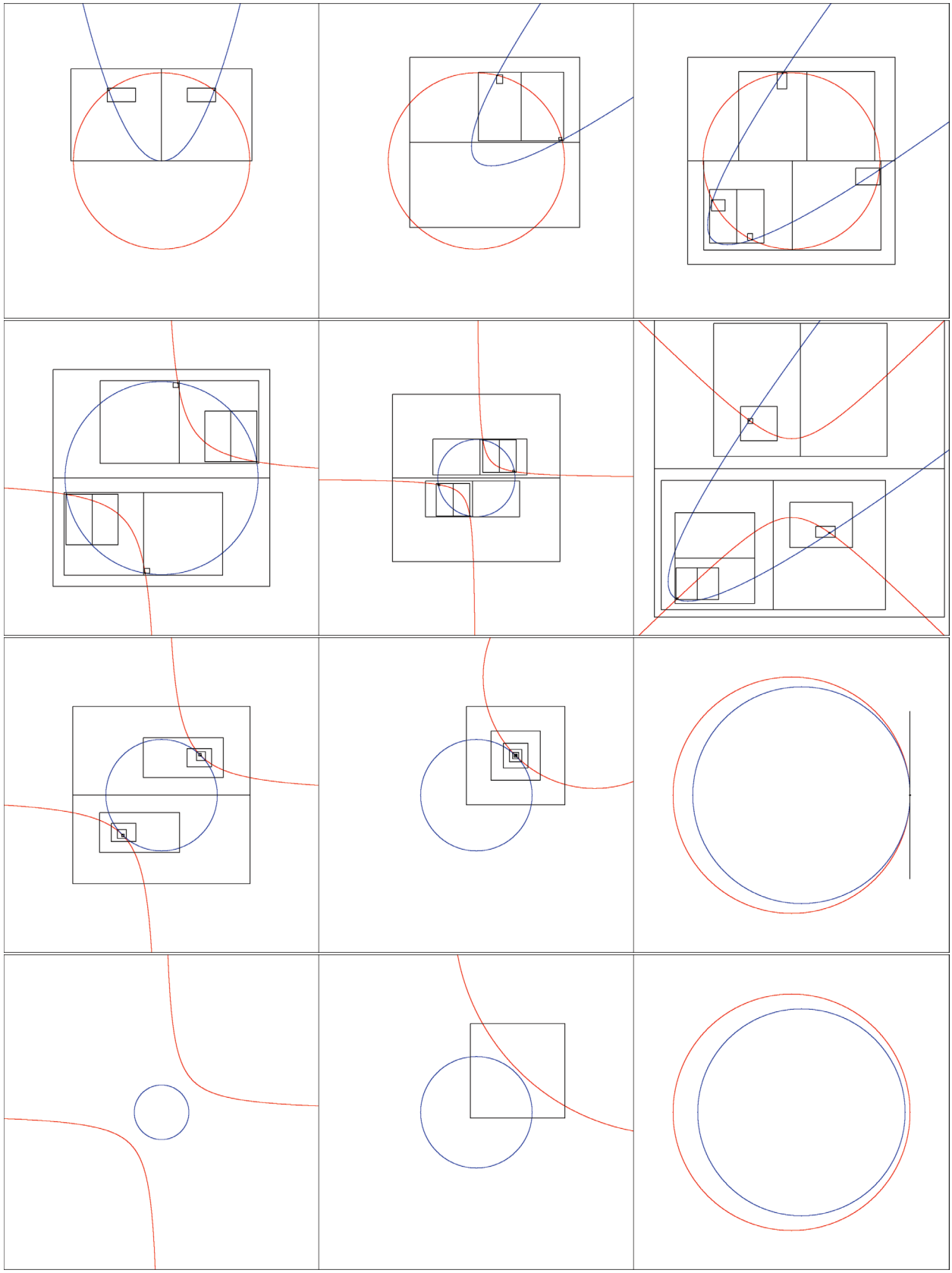


Figure 6: Each figure shows all the domain boxes, which are reduced or bisected by our solver using LP reductions. First two rows: the convergence is super-quadratic around a non-singular solution. Third row: the convergence is linear for near-singular points of intersection (tangential contact between two curves), but each side length is divided by more than two: otherwise, the interval is bisected. Last row: the boxes without roots are quickly detected, and one or two reductions prove them empty.

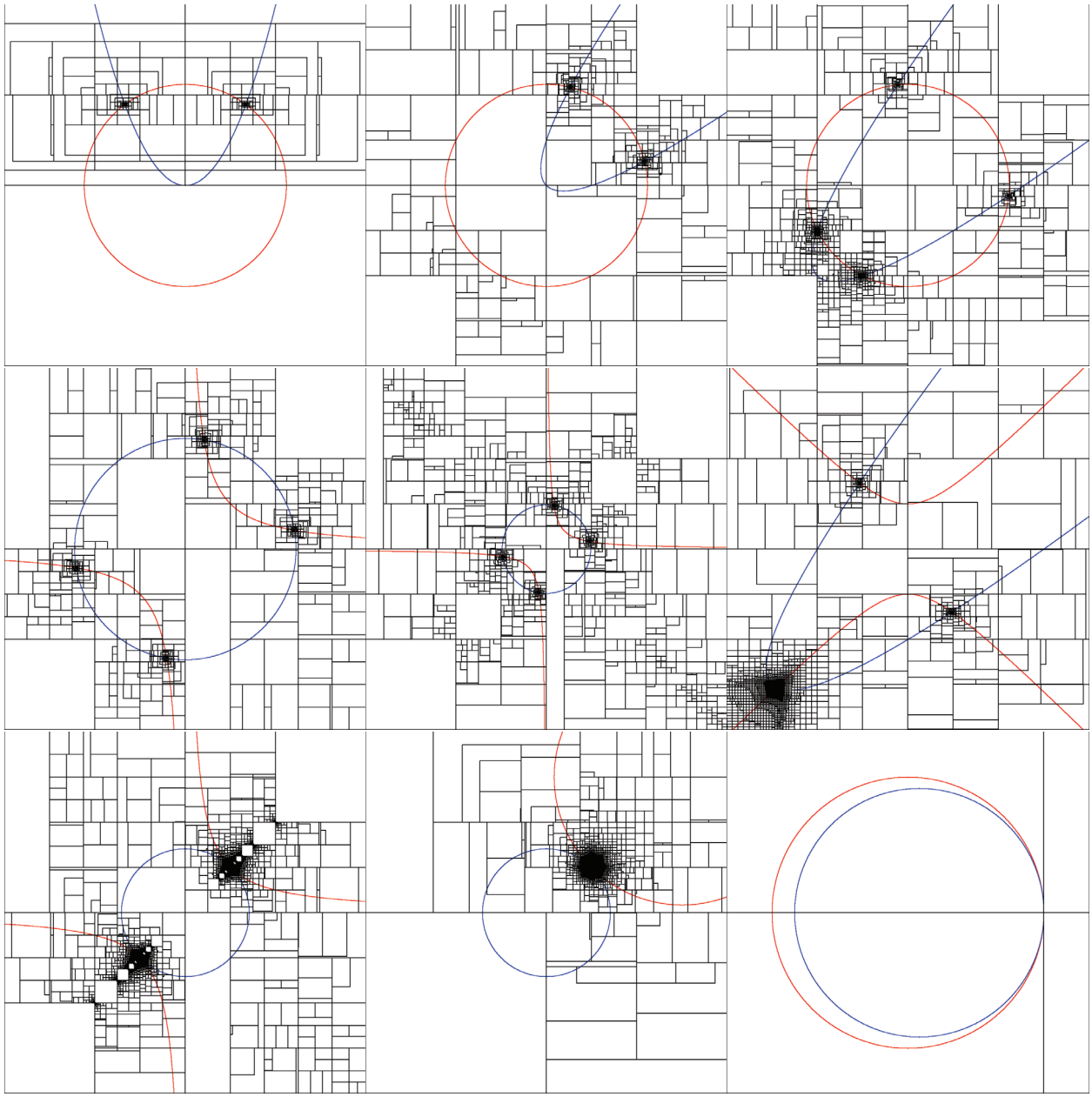


Figure 7: Each figure shows all the domain boxes, which are reduced or bisected by a solver using preconditioning and reductions with standard interval arithmetic.

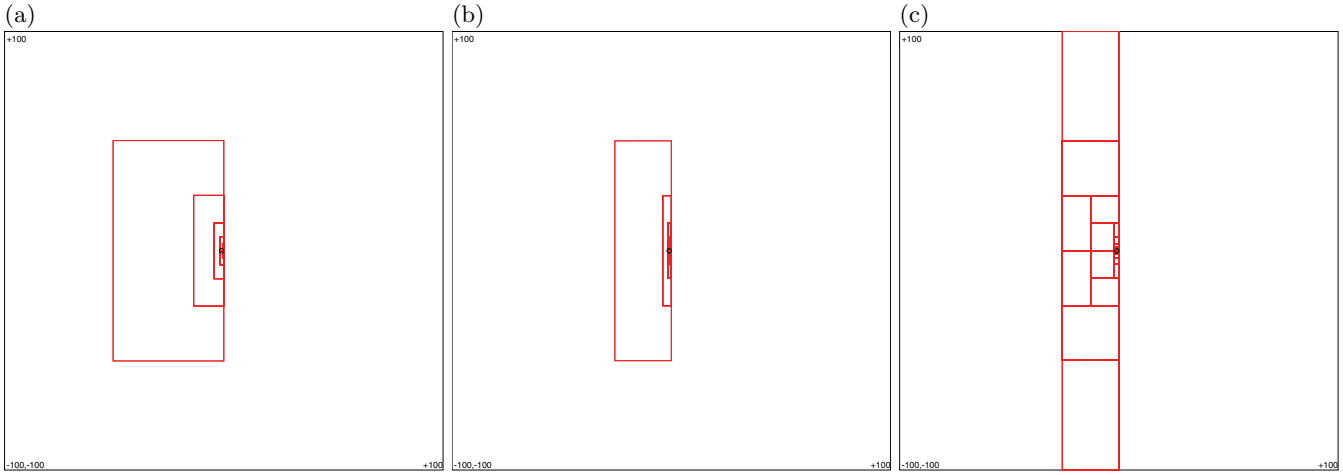


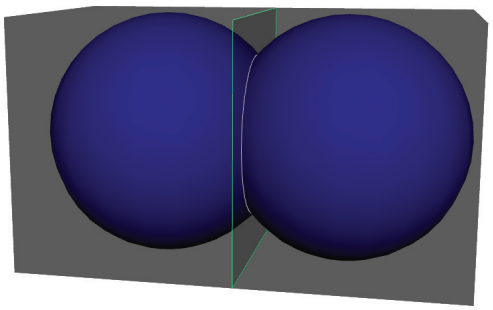
Figure 9: Domain boxes in the x_0, y_0-plane with different reduction/bisection orders: (a) Reduce all dimensions once before bisecting the largest dimension, (b) Reduce all dimensions as long as effective then bisect the largest dimension, (c) Reduce only the largest dimension as long as effective then bisect it.

5.3 Sphere-Sphere Surface Intersection (3D)

This section considers an under-constrained system, i.e., with less equations than unknowns. A computation is still possible but the solution is not a discrete set anymore. Such a system is given by the intersection of the surfaces of two spheres (centers $(0,0,0)$ and $(3/4,0,0)$ and radii $1/2$) in 3-space (Figure 10):

$$\begin{aligned} f_1(x,y,z) &= x^2+y^2+z^2-1/4 \\ f_2(x,y,z) &= x^2+y^2+z^2-3/2x+5/16 \end{aligned} \tag{11}$$

The solution set is covered by 3072 boxes of interval width $\delta = 10^{-3}$ in 5367 reductions and 3071 subdivisions (reduction/bisection order: reduce all once). Figure 10 also shows the domain boxes analyzed in the y,z-plane. The solution time is 1.52 seconds for this problem, which was also considered in the paper [16] about a barycentric Bernstein basis solver, and the subdivision statistics are similar. Unfortunately, the runtime is not available in [16].



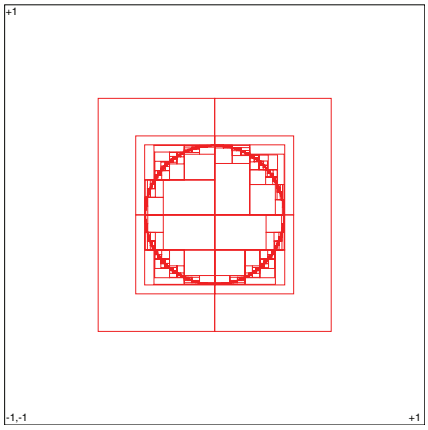


Figure 10: Intersection of two spheres with radii $r = 0.5$ and centers in $(0,0,0)$ and $(0.75,0,0)$.

5.4 Gough-Stewart Platform

The Gough-Stewart platform is used as a parallel robot. It is a structure made of two triangles connected by jacks (translational joints) into an octahedron [18]. See Figure 11 for an illustration.

The lower triangle serves as the base, and the upper triangle moves as the work platform. Edges of the platform and of the base are rigid, i.e., their lengths are fixed once (brown triangles in Figure 11): p_2p_3, p_3p_1, p_1p_2, p_6p_4, p_4p_5, p_5p_6. The lengths of the remaining edges are computer-controlled (gray lines in Figure 11): p_1p_4, p_1p_5, p_2p_5, p_2p_6, p_3p_6, p_3p_4. The Cayley-Menger determinant gives a relation of the distances between 5 points in 3-space [13].

$$\begin{aligned} \det(M) &= \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12} & d_{13} & d_{14} & d_{15} \\ 1 & d_{21} & 0 & d_{23} & d_{24} & d_{25} \\ 1 & d_{31} & d_{32} & 0 & d_{34} & d_{35} \\ 1 & d_{41} & d_{42} & d_{43} & 0 & d_{45} \\ 1 & d_{51} & d_{52} & d_{53} & d_{54} & 0 \end{pmatrix} = 0 \\ &= a_9 d_{24}^2 d_{35}^2 + a_8 d_{24}^2 d_{35} + a_7 d_{24} d_{35}^2 \\ &\quad + a_6 d_{24}^2 + a_5 d_{35}^2 + a_4 d_{24} d_{35} \\ &\quad + a_3 d_{24} + a_2 d_{35} + a_1 \end{aligned} \tag{12}$$

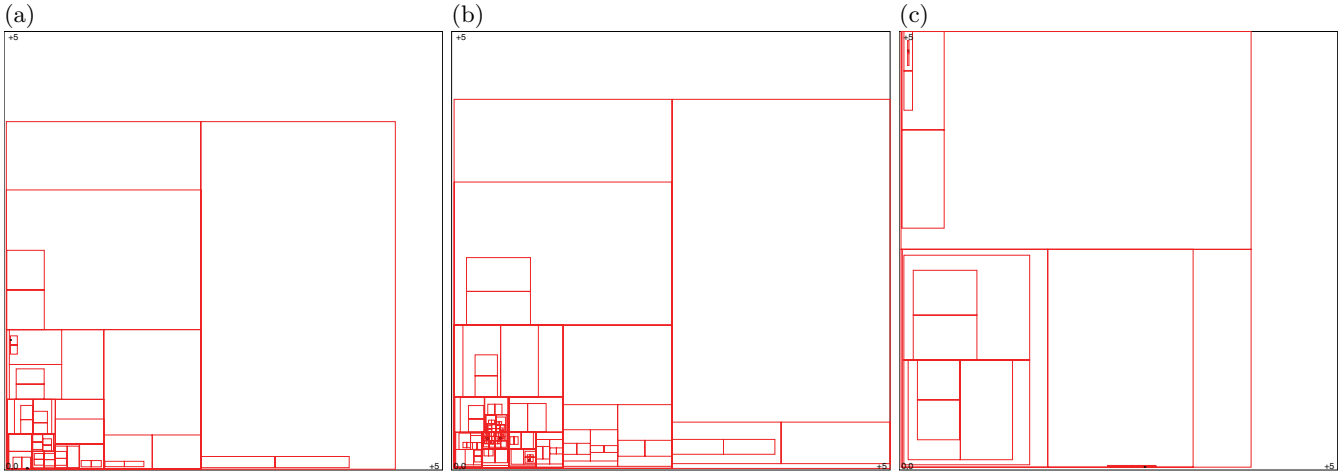


Figure 12: Solving Cayley-Menger equations for d_{24} (horizontal) and d_{35} (vertical). There are (a) 2, (b) 3, (c) 2 positive, real solutions found, marked by small black boxes.

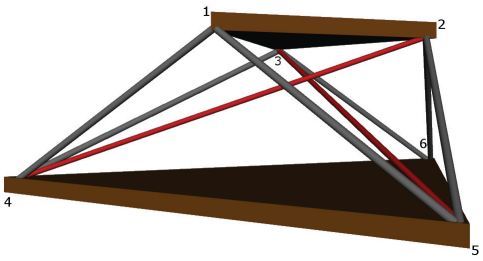


Figure 11: Gough-Stewart platform built from an octahedron. The red lines show the two diagonals between non-adjacent vertices, whose squared lengths are used as variables in the Cayley-Menger formulation.

For points p_1, p_2, p_3, p_4, p_5, the Cayley-Menger determinant incorporates the squared distances d_{12}, d_{13}, d_{14}, d_{15}, d_{21}, d_{23}, d_{24}, d_{25}, d_{34}, d_{35}, d_{45}. All these distances are known, except for d_{24} and d_{35} (red lines in Figure 11), and the equation has degree 4 in 2 unknowns. A second, independent equation can be generated for the points p_6, p_2, p_3, p_4, p_5. We can solve this system of degree 4 by substituting an auxilliary variable $d_{24,35}$ for the product $d_{24,35} := d_{24}d_{35}$. Then the monomials $d_{24}^2 d_{35}^2$, $d_{24}^2 d_{35}$, and $d_{24} d_{35}^2$ become degree at most two: $d_{24,35}^2$, $d_{24}d_{24,35}$, and $d_{24,35}d_{35}$.

Figure 12 shows all domain boxes during solving these systems with minimum interval width $\delta = 10^{-4}$, and it shows all positive, real results.

Once the lengths d_{24} and d_{35} of two diagonals are known, we can compute consistent coordinates for the six vertices. So the *forward kinematics* of the Stewart platform is fully determined by the lengths of two diagonals between non-adjacent vertices.

5.5 Circle Packing

Nice and increasingly complex geometric constraint solving problems can be generated from circle packing in the plane. Given a planar graph on n vertices, compute a set of circle centers corresponding to the graph's vertices and their radii so that circles of adjacent vertices are tangent. In order to make for a unique solution, we have to add further conditions: the planar graph is fully triangulated, and the coordinates of three arbitrary centers are fixed. The latter is necessary to avoid affine motions and inversions in the plane.

Let G be the planar, triangulated graph with n vertices, and call c_i the center and r_i the radius of vertex $i \in V(G)$. Then the following system formulates circles' tangency

$$(c_i - c_j)^t (c_i - c_j) \quad = \quad (r_i + r_j)^2 \text{ if } \{i,j\} \in E(G) \qquad (13)$$

and the following triangle-inequality constraints prohibit circles' overlap

$$(c_i - c_j)^t (c_i - c_j) \quad \geq \quad \begin{array}{r} (r_i + r_j)^2 \text{ if } \{i,j\} \notin E(G) \\ \text{but } \{i,k\}, \{k,j\} \in E(G) \end{array} \qquad (14)$$

If the planar graph G is triangulated except for the outer face then a continuous set of solutions exists, as we show in Figure 13 for $n = 4$. If we omit inequalities (14) a discrete set of distinct solutions is found by the solver for $n \geq 5$, in which circles possibly fall together. Figure 14 shows the unique solution of a circle-packing problem with $n = 12$ circles and tangencies taken from an icosahedron. The solution times for problem instances from $n = 4$ to $n = 20$ (planar graphs randomly generated) are given in Figure 15 for minimum interval width $\delta = 10^{-2}$. We have used the following reduction/bisection order: reduce all dimensions once before bisecting the largest dimension. The graph compares runtime using only the Bernstein polytope with runtime, where the solver switches to thick hyperplanes as described in Section 4. In all these instances, we start with the box $([0,30] \times [0,30] \times [0,10])^n$, the outer triangle fixed to coordinates $(10,10)$, $(20,10)$, $(15,20)$, and the largest radius

being 6.1803. The corresponding number of reductions and bisections is contained in Figure 16 and in Figure 17 with eventual switches to thick hyperplanes. The overall effect of switches to thick hyperplanes is small for $\delta = 10^{-2}$, as they are only used in a few reductions. But for smaller interval widths ϵ, they occur ever more often.

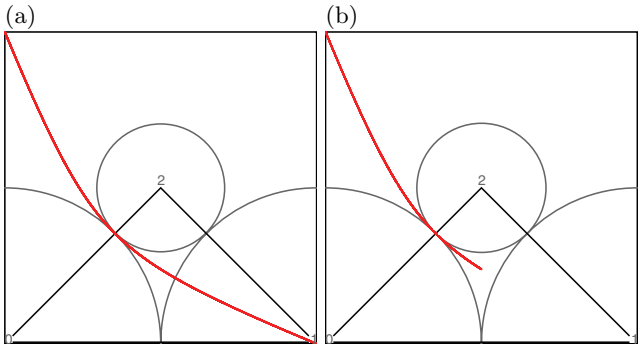


Figure 13: (a) Solution of the circle-packing problem with only equations (13) for a triangulated quadrangle. The trace of red boxes is the solution set for the center of the fourth circle tangent to circles 0 and 2. (b) A non-overlap constraint (14) of circle 3 and 1 has been added to the system in (a).

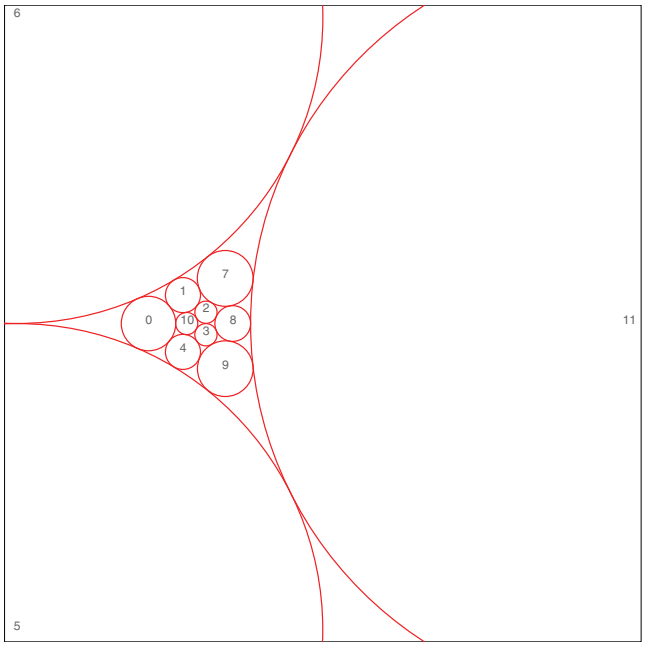


Figure 14: Solution of a circle-packing problem with $n = 12$ circles with tangencies taken from an icosahedron.

6. CONCLUSION AND FUTURE WORK

This paper presents a nonlinear subdivision solver using polynomial time domain reductions. It overcomes the difficulties due to the exponential cost of representing multivariate polynomials in the TBB. It instead uses a Bernstein polytope described by a polynomial number of halfspaces. A domain interval for the given system can be reduced by a

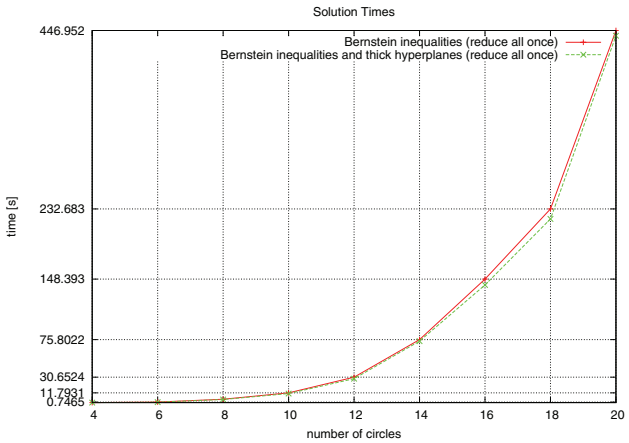


Figure 15: Solution times for the circle-packing problem of different problem sizes.

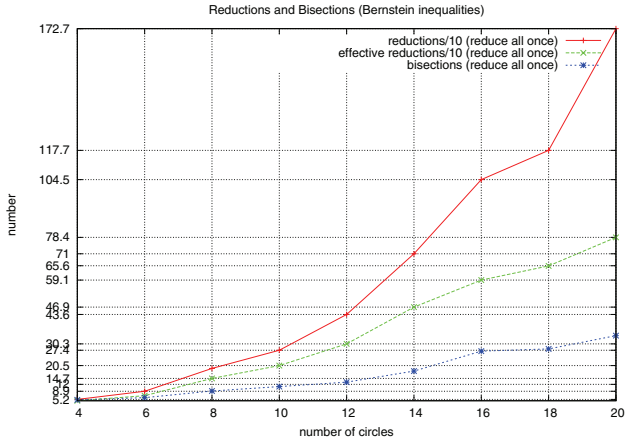


Figure 16: Number of reductions and bisections during circle-packing when using Bernstein inequalities only.

standard LP solver using a linearization of the system with the Bernstein polytope. Due to floating point inaccuracy, it

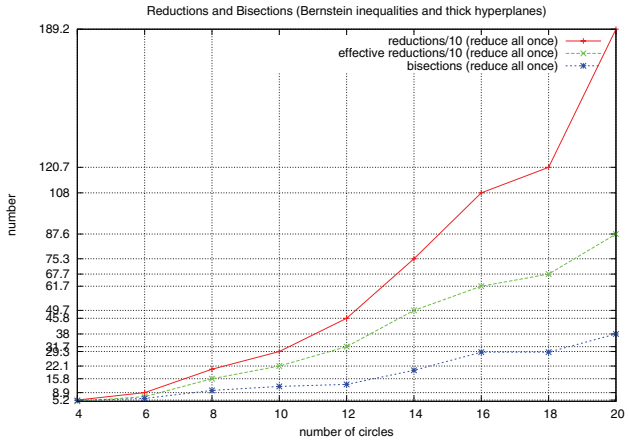


Figure 17: Number of reductions and bisections during circle-packing when using thick hyperplanes for flat Bernstein hulls.

is necessary to modify the Bernstein polytope's inequalities after a reduction in order to not omit solutions.

The proposed solver does not need any special interval LP solver for a robust and efficient implementation. The runtime behavior can be optimized in several ways, either by parallelizing reductions (GPU) or by special techniques in the LP solver (primal-dual simplex algorithms, generating good start bases, exploiting the special form of the Bernstein polytopes).

As future work, we would like to consider higher order systems, transcendental systems and connected solution sets. The Bernstein polytope can be generalized to higher degrees but it is also possible to reduce higher degree systems to quadratic systems by symbolic substitution. If the polynomial is represented as a binary tree, each leaf node carries a numerical constant or an unknown x_i, and each inner node carries an arithmetic operation $\{+, -, \times\}$ together with two subtrees. Adding a new variable z for each inner node allows to construct a quadratic system. If $o \in \{+, -, \times\}$ is the operation carried out by the node and x, y are the variables or constants of the two child nodes then the equation defining variable z is $z = x\ o\ y$. The result is either a linear equation or a quadratic equation.

The solver may be generalized to equations containing trigonometric or exponential functions. It suffices to generate an enclosing polytope for the curve, for example $(x, \cos x)$, $x \in [u, v]$. This possibility is a big advantage compared to other solvers such as homotopy.

Recently, several methods have been proposed to approximate continuous sets defined by systems of equations and inequalities, while guaranteeing that the approximation and the exact object have the same topology [19]. These methods do not handle objects defined by projections. We will investigate the possibility to adapt the solver to deal with these problem types.

7. REFERENCES

[1] B. Bruderlin and D. Roller, editors. *Geometric Constraint Solving and Applications*. Springer, 1998.

[2] V. Chvatal. *Linear Programming (Series of Books in the Mathematical Sciences)*. W. H. Freeman, September 1983.

[3] I. Duff, A. Erisman, and J. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986.

[4] C. B. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Purdue University, 1998.

[5] G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *SMA'01: Proc. of the 6th ACM Symp. on Solid Modeling and Applications*, pages 1–10, New York, NY, USA, 2001. ACM Press.

[6] G. Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press Professional, San Diego, California, 1988.

[7] J. J. H. Forrest and J. A. Tomlin. Updating triangular factors of the basis to maintain sparsity in the product form simplex method. *Mathematical Programming*, (2):263–278, 1972.

[8] T. Grandine. Geometry processing (chapter 24). In *Handbook of CAGD (Farin, Hoschek, Kim eds.)*, pages 603–623. Elsevier, 2002.

[9] C. Hoffmann and B. Yuan. There are 12 common tangents to four spheres. 2000. `http://www.cs.purdue.edu/homes/cmh/distribution/SphereTangents.htm`.

[10] R. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, Netherlands, 1996.

[11] H. Lamure and D. Michelucci. Solving constraints by homotopy. In *Proc. of the Symp. on Solid Modeling Foundations and CAD/CAM Applications*, pages 263–269, May 1995.

[12] D. Michelucci. Linear Programming for Interval Newton Solvers - Extended abstract on a work in progress. In *Automatic Deduction in Geometry, ADG 2008, Shanghai*, 2008.

[13] D. Michelucci and S. Foufou. Using Cayley-Menger determinants for geometric constraint solving. In *SM '04: Proceedings of the ninth ACM Symposium on Solid Modeling and Applications*, pages 285–290, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[14] D. Michelucci and S. Foufou. Bernstein basis for interval analysis: application to geometric constraints systems solving. *8th Conference on Real Numbers and Computers (Bruguera and Daumas, eds.)*, pages 37–46, July 2008.

[15] B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report RR-5658, INRIA, August 2005.

[16] M. Reuter, T. S. Mikkelsen, E. C. Sherbrooke, T. Maekawa, and N. M. Patrikalakis. Solving nonlinear polynomial systems in the barycentric Bernstein basis. *The Visual Computer*, 24(3):187–200, 2008.

[17] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Comput. Aided Geom. Des.*, 10(5):379–405, 1993.

[18] D. Stewart. A Platform with Six Degrees of Freedom. *UK Institution of Mechanical Engineers Proceedings*, 180 Part 1(15):371–386, 1965-66.

[19] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha. Topology preserving isosurface extraction for geometry processing. In *Second Eurographics Symposium on Geometry Processing*, pages 235–244, 2004.

[20] C. Wampler, A. Morgan, and A. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. on Design*, (112):59–68, 1990.

[21] J. Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, 1965.

[22] R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, TU Berlin, 1996.